# *Software Requirements Specification*

***for***

# *COSC Club Event Management System*

**Prepared by**


| | | |
|---|---|---|
| **Maria DeMelo** | **199695520** | **mdemelo@algomau.ca** |
| **Emma Ude** | **199697810** | **cude@algomau.ca** |
| **Erfan Razavi** | **199658710** | **srazavi@algomau.ca** |
| **Jason Guo** | **199666070** | **yguo@algomau.ca** |


**Instructor:** ***Amandeep S. Patti***

**Course:** **COSC 3506**

**Date:** **Feb 11, 2022**

# Contents

# 1 Introduction

## 1.1 Document Purpose

*The purpose of this document is to present a detailed description of the COSC Club Event Management System. It provides crucial information regarding the product and its goal. Thus, this document describes the functionality, target audience and requirements of the system.*

## 1.2 Product Scope

*The scope of this project will be to provide a system that allows the CCMT to:*
*1. manage the undergraduate COSC clubs*
*2. manage undergraduate events : create and cancel events*
*3. manage financial transactions of these undergraduate COSC clubs*
*4. manage system users*
*5. manage the memberships of undergraduate students*

## 1.3 Intended Audience and Document Overview

*This document is intended for the CCMT, the Professor and the software developers. After the introduction, the document goes into a brief description of the software product and its functionalities in Section 2, as well as the design constraints and other factors involved in the development of the product. Then in Section 3 the functionalities and the user interface are described in depth, including a use case diagram and its description. In Section 4, the non-functional requirements are stated and described in depth.*

## 1.4 Definitions, Acronyms and Abbreviations

*Below is a table of all the abbreviations and acronyms used in this SRS document:*

| | |
|---|---|
| *CCMT* | *COSC Club Management Team* |
| *CCEMS* | *COSC Club Event Management System* |
| *COSC* | *Computer Science* |
| *SRS* | *Software Requirement Specification* |
| *UCL* | *Undergraduate Club Leader* |

## *1.5 Document Conventions*

### *1.5.1 Text Formatting Conventions*

*This document follows the IEEE formatting requirements. The entire document uses Arial font. The font size for the text is 11 and italicized. The text uses Justify alignment and is single spaced. A 1" margin is maintained throughout this document.*

### *1.5.2 Heading Formatting Conventions*

*The headings are divided into Sections and subsections. The section headings have a font size of 18 and the subsection headings have a font size of 14 or 12. The headings are bolded with the section headings in white and the subsection headings in black. The section headings are center-aligned and the subsection headings are left-aligned.*

# 2 Overall Description

## 2.1 Product Overview

*The CCMT is in need of a software that allows for better management of their club. Currently it is extremely hard for the organizers to keep track of events, memberships and the financial information. The application to be created is meant to organize and facilitate the leaders with a better management tool which consequently decreases potential errors. The software product being created is a new, self-contained product. This SRS defines the functionality requirements of the entire CCMES.*


*Figure 1: Product Overview*

## 2.2 Product Functionality

*The system is supposed to perform the following major functions:*
*1. Login to the system.*
*2. Manage memberships of undergraduate students*
*3. Manage undergraduate club events*
*4. Manage club finances*

## 2.3 Design and Implementation Constraints

*Implementation constraints are limitations that have the potential to hinder our progress in building the application. Since the project must be complete by April 1st, we are limited on time that could have otherwise been used to further improve the application. Another potential constraint is that we are only able to use Java and JavaFX when designing the software. This limits the tools we have to expand the application and improve its overall functionality. We are using the COMET method for software design which begins with requirements and analysis modeling. Next it moves through the design modelling stage and once that is complete it moves to the construction and implementation*

*stages. Finally the system is tested to ensure everything is running smoothly. At any point, the software can recycle previous steps to fix, change or update.*

## *2.4 Assumptions and Dependencies*

*Assumptions and dependencies give insight about tools that are assumed the user has. For example, since emails are having to be sent, it must be assumed that the user will have access to internet connection. Further, since data will need to be entered into the program, it must be assumed that the user can input the data. Because the user is having to use Java, it's assumed that they have a JDK installed on their personal device and are somewhat familiar with using the Java program.*

# 3 Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

*The user interface will be designed as an embedded system, where the user can move from one page to another. The user does this by clicking on an icon or button that will lead them to a different page. We will have about ten distinct pages. First the user logs in and is taken to the mainmenu where they can navigate to the Event page, Finance page, or Member page. Each of these pages lead to other pages (create, withdraw, add, etc) that are specific to their functionality. Below is a basic diagram of how the interface will look like.*

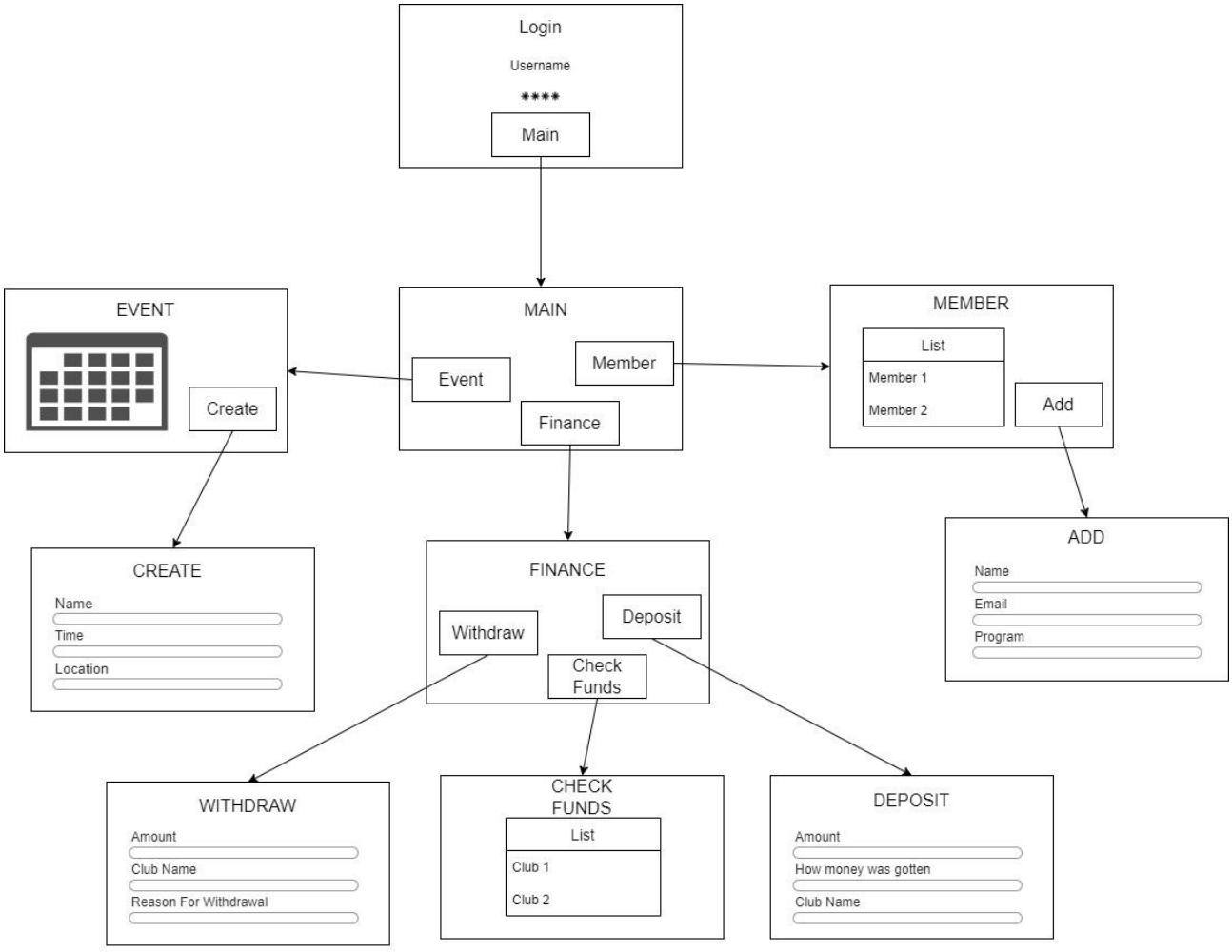


*Figure 2: User interface*

### *3.1.2 Hardware Interfaces*

*No hardware interface will be created for this product as it is a software-intensive product. It will be created to be used on a laptop or desktop computer.*

### *3.1.3 Software Interfaces*

*This product will be made using Java and JavaFX programming languages. This was stated as one of the requirements stipulated by the client.*

## *3.2 Functional Requirements*

### *3.2.1 F1: Login to the system*

*This requirement allows only the intended users of the system to be able to interact with it. It is very important and is the first step in accessing the systems. This functionality involves entering a username and password. The password will also have to be verified by the system.*

### *3.2.2 F2: Manage Membership of Undergraduate Students*

*There will be a list of various members of the different COSC clubs. This list includes basic information on the students, such as name, program of study, years spent in the club, club positions, student email etc. The users will be able to add and remove members from the list as necessary. Each UCL will only be granted access to their club's information. However, the CCMT will have access to all the members of all the COSC clubs.*

### *3.2.3 F3: Manage Undergraduate Club Events*

*Users of the system will be able to create and cancel events. UCLs can only cancel the events that they created personally, however, the CCMT can cancel all events. When an event is created, there will be an option to advertise that event to the club members via their email.*

### *3.2.4 F4: Manage Club Finances*

*The UCLs can only access the finance information of their specific club. The CCMT has access to all the financial information of every club. The UCLs can check the funds allotted to their club. They can also request to withdraw money from the club funds and if there is sufficient balance they will be allowed to make the request to the CCMT. They can also document the deposits gotten from things like fundraisers.*

## *3.3 Use Case Model*

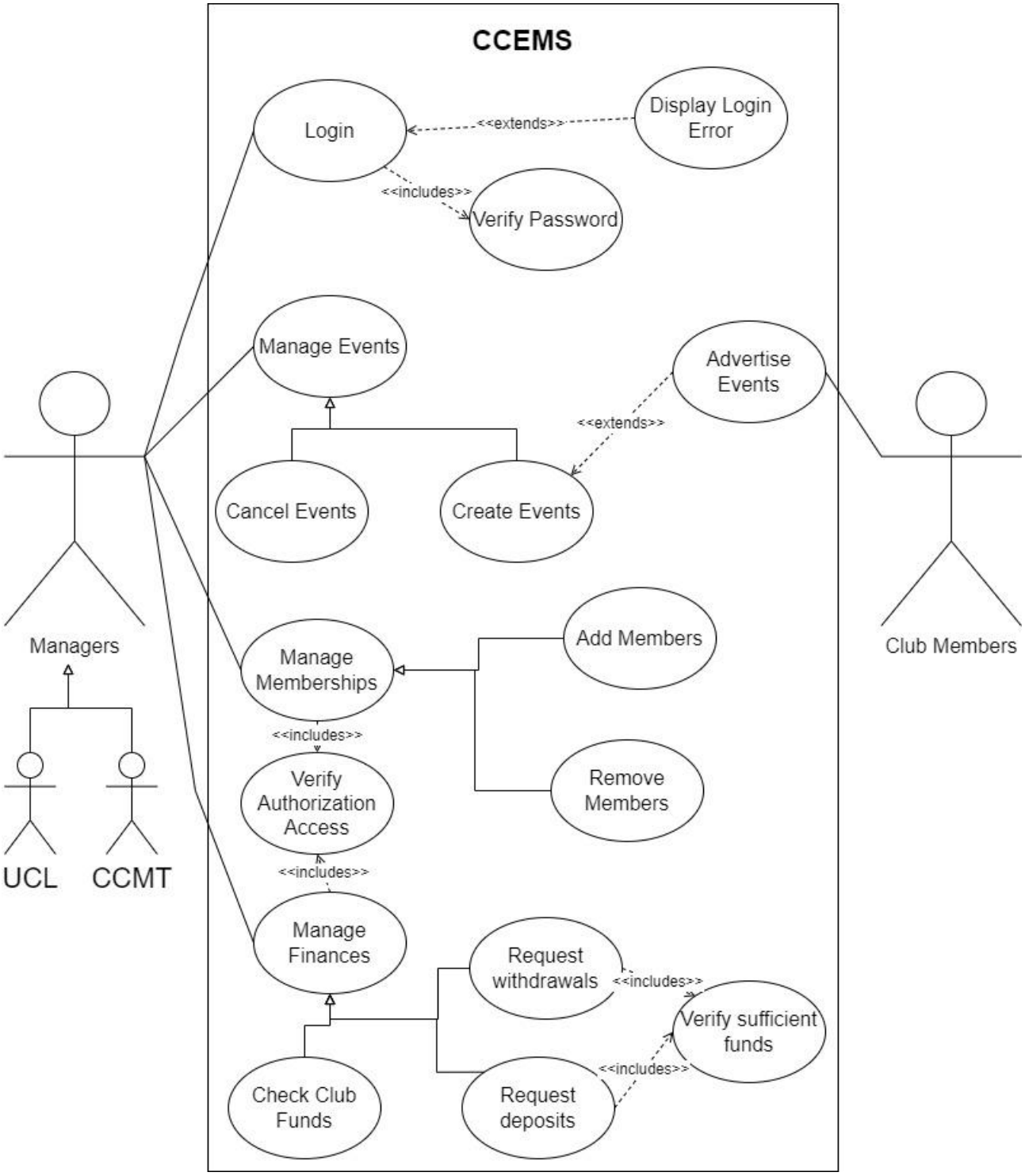


*Figure 3: Use Case Diagram*

### *3.3.1 U1: Login*

***Author –*** *Emma Ude*

***Purpose*** *- This use case describes the ability for users to login.*

***Requirements Traceability –*** *This can be traced to functionality requirement F1*

***Priority*** *- High. This use case is necessary for the user to be able to interact with the system*

***Preconditions*** *- There are no preconditions*

***Post conditions*** *- The user will be logged in and can access the system*

***Actors*** *– Managers*

***Extends –*** *This is not an extension use case*

***Flow of Events***

1. *Basic Flow - The user logs in with their username and password and then the system verifies that password and grants them access to the system.*
2. *Alternative Flow - if the user's login information is incorrect, a login information is incorrect, then the system will display an error message and the user will not be granted access.*
3. *Exceptions - there are no exceptions*

***Includes*** *- U2*

### *3.3.2 U2 : Verify Password*

***Author –*** *Emma Ude*

***Purpose*** *- To ensure that the user entered the correct password before granting them access to the system.*

***Requirements Traceability –*** *It can be traced to F1*

***Priority*** *- High, necessary to gain access.*

***Preconditions*** *- The user must first attempt to login to the system.*

***Post conditions*** *- The user will be logged in and can access the system*

***Actors*** *– no actors interact with this use case*

***Extends –*** *none.*

***Flow of Events***

1. *Basic Flow - the system checks if the password is valid, and lets the user access the system*
2. *Alternative Flow - if the password is invalid, an error message will be displayed*
3. *Exceptions - Exceptions that may happen during the execution of the use case*

***Includes -*** *none*

***Notes/Issues*** *- This use case is included in U1*

### *3.3.3 U3: Display Login Error*

***Author –*** *Emma Ude*

***Purpose*** *- Display an error message if the login was not successful and deny the user access*

***Requirements Traceability –*** *Traced to requirement F1*

***Priority*** *- High, this use case is important to ensure the security of the system*

***Preconditions*** *- The login information (username or password) has to be incorrect*

***Post conditions*** *- The user is denied access to the system*

***Actors*** *– no actors interact with this use case*

***Extends –*** *Extends U1*

***Flow of Events***

1. *Basic Flow - the user is not granted access to the system*
2. *Alternative Flow - no other alternative*
3. *Exceptions - no exceptions*

***Includes -*** *none*

### *3.3.4 U4: Manage Events*

***Author –*** *Emma Ude*

***Purpose*** *- To allow the users to manage events. This includes access to future events that have been scheduled, and who is to attend these events. It also includes past events that have been held and the attendees and hosts.*

***Requirements Traceability –*** *Can be traced to requirement F3*

***Priority*** *- High, it is important that this use case works properly as it is the main purpose of having this application*

***Preconditions*** *- no preconditions*

***Post conditions*** *- The user will have made changes to existing events or created new events.*

***Actors*** *– Managers*

***Extends –*** *none*

***Flow of Events***

1. *Basic Flow - The user will see past and future events that have been scheduled. They will be given the opportunity to modify these events.*
2. *Alternative Flow - no alternative*
3. *Exceptions - no exceptions*

***Includes*** *- none*

***Notes/Issues*** *- This is a parent use case. It has two children use cases: U5 and U6*

### *3.3.5 U5: Cancel Events*

***Author –*** *Emma Ude*

***Purpose*** *- To cancel existing events*

***Requirements Traceability –*** *Traceable to requirement F3*

***Priority*** *- Medium, If there are issues this use case will not affect the overall functionality of the system. Although it should be maintained as soon as possible to avoid confusion.*

***Preconditions*** *- The event to be deleted must already exist.*

***Post conditions*** *- The chosen event will be deleted*

***Actors*** *– Managers*

***Extends –*** *none*

***Flow of Events***

1. *Basic Flow - the chosen event is deleted*
2. *Alternative Flow - if it is a UCL, they can only delete an event they created*
3. *Exceptions - cannot delete a past event.*

***Includes*** *- none*

***Notes/Issues*** *- This is a child use case of U4*

### *3.3.6 U6: Create Events*

***Author –*** *Emma Ude*

***Purpose*** *- create a new event.*

***Requirements Traceability –*** *Traceable to requirement F3*

***Priority*** *- High, this use case is a necessary function of the system*

***Preconditions*** *- no preconditions*

***Post conditions*** *- A new event will be created*

***Actors*** *– Managers*

***Extends –*** *none*

***Flow of Events***

1. *Basic Flow - The user fills in the information required to create a new event such as the name of the event, time , location, etc. The event is then created and added to the list of existing events*
2. *Alternative Flow - if the user chooses to, the event can be advertised.*
3. *Exceptions - an event cannot be created in the past.*

***Includes*** *- none*

***Notes/Issues*** *- this is a child use case of U4*

### *3.3.7 U7: Advertise Events*

***Author –*** *Emma Ude*

***Purpose*** *- To advertise an event via email to the students that are expected to attend.*

***Requirements Traceability –*** *Traceable to requirement F3*

***Priority*** *- Medium, since it is an optional function.*

***Preconditions*** *- The event to be advertised must already exist and the user must express interest in advertising their event*

***Post conditions*** *- An email will be sent to the students listed as attendees for the event*

***Actors*** *– Managers send an email. Club Members receive an email from this use case.*

***Extends –*** *U6*

***Flow of Events***

1. *Basic Flow - the user can send an email to the attendees about their event.*
2. *Alternative Flow - if the user does not want to advertise the event then no email will be sent.*
3. *Exceptions - no exceptions*

***Includes*** *- none*

### *3.3.8 U8: Manage Memberships*

***Author –*** *Emma Ude*

***Purpose*** *- To be able to view the list of current members of the clubs and modify that list.*

***Requirements Traceability –*** *Traceable to requirement F2*

***Priority*** *- High, it contains the personal information of the students in the clubs and is sensitive information.*

***Preconditions*** *- no preconditions*

***Post conditions*** *- The user will have accessed the list of members and may have modified that list*

***Actors*** *– Managers*

***Extends –*** *none*

***Flow of Events***

1. *Basic Flow - the user is given access to the list of members and allowed to modify the list*
2. *Alternative Flow - if the user has specific authorization, they can only access the list of specific members*
3. *Exceptions - no exception*

***Includes*** *- U11*

***Notes/Issues*** *- This is a parent use case and contains the child use cases: U9 and U10*

### *3.3.9 U9: Add Members*

***Author –*** *Emma Ude*

***Purpose*** *- To add new members to the list of members*

***Requirements Traceability –*** *Traceable to requirement F2*

***Priority*** *- High, the users should be able to add new member,*

***Preconditions*** *- The member must not already exist in the list*

***Post conditions*** *- A new member will be added to the list*

***Actors*** *– Managers*

***Extends –*** *none*

***Flow of Events***

1. *Basic Flow - the user fills in the information for the new member and the member's information is added to the list of existing members*
2. *Alternative Flow - no alternative*
3. *Exceptions - the member cannot already exist*

***Includes*** *- none*

***Notes/Issues*** *- This is a child use case of U8*

### *3.3.10 U10: Remove Members*

***Author –*** *Emma Ude*

***Purpose*** *- To remove existing members from the list*

***Requirements Traceability –*** *Traceable to requirement F2*

***Priority*** *- Medium, it is not an urgent function, but should be maintained as soon as possible*

***Preconditions*** *- The member must already exist*

***Post conditions*** *- The member's information will be deleted from the list*

***Actors*** *– Managers*

***Extends –*** *none*

***Flow of Events***

1. *Basic Flow - the members information will be deleted from the list*
2. *Alternative Flow - no alternative*
3. *Exceptions - The member must already exist*

***Includes*** *- none*

***Notes/Issues*** *- This is a child use case of U8*

### *3.3.11 U11: Verify Authorization Access*

***Author –*** *Emma Ude*

***Purpose*** *- To verify if a user has access to certain information*

***Requirements Traceability –*** *Traceable to F2*

***Priority*** *- High, ity is necessary to maintain privacy of information*

***Preconditions*** *- The user must have started use case U8 or U12*

***Post conditions*** *- The user will be granted access to the information that is at their authorization level*

***Actors*** *– no actors interact with this use case*

***Extends –*** *none*

***Flow of Events***

1. *Basic Flow - The system checks the login information of the current user. if the user is a UCL, they will be granted access to information that is specific to the club they*

*lead. If they are part of the CCMT, they will have access to all the information available.*
2. *Alternative Flow - no alternative*
3. *Exceptions - no exceptions*

***Includes*** *- none*

***Notes/Issues*** *- This is a use case that is included in U8 and U12*

### *3.3.12 U12: Manage Finances*

***Author –*** *Emma Ude*

***Purpose*** *- To see a detailed description of the current money available to a club and the past transactions done. Also the ability to document new transactions*

***Requirements Traceability –*** *Traceable to requirement F4*

***Priority*** *- High, this use case contains sensitive information.*

***Preconditions*** *- no preconditions*

***Post conditions*** *- The user will have access to the financial information and may have added new information*

***Actors*** *– Managers*

***Extends –*** *none*

***Flow of Events***

*Basic Flow - the user will be given access to the financial information, and will be allowed to add more information.*
*Alternative Flow - if the user has specific authorization, they can only access specific financial information, not all of it.*
*Exceptions - no exceptions.*

***Includes*** *- U11*

***Notes/Issues*** *- This is a parent use case with three child use cases: U13, U14, and U15*

### *3.3.13 U13: Check Club Funds*

***Author –*** *Emma Ude*

***Purpose*** *- To check the funds allotted to a specific club*

***Requirements Traceability –*** *Traceable to requirement F4*

***Priority*** *- High, contains sensitive information*

***Preconditions*** *- must have access to that specific club's information.*

***Post conditions*** *- The user will see what funds the club currently has.*

***Actors*** *– Managers*

***Extends –*** *none*

***Flow of Events***

*Basic Flow - the user will see the funds allotted to the clubs*
*Alternative Flow - if the user belongs to a specific club(a UCL), they can only access their club's funds*
*Exceptions - no exceptions*

***Includes*** *- none*

***Notes/Issues*** *- This is a child use case of U12*

### *3.3.14 U14: Request Withdrawal*

***Author –*** *Emma Ude*

***Purpose*** *- To request to withdraw money from a specific club's financial account*

***Requirements Traceability –*** *Traceable to requirement F4*

***Priority*** *- High, contains sensitive information*

***Preconditions*** *- must have access to that specific club's information.*

***Post conditions*** *- The user's request will be documented in the financial information of that club*

***Actors*** *– Managers*

***Extends –*** *none*

***Flow of Events***

*Basic Flow - the user will provide information such as, how much they are withdrawing, what the money is to be used for, who is making the withdrawal, etc. Then the request is documented.*
*Alternative Flow - if the user belongs to a specific club(a UCL), they can only request funds from that specific club*
*Exceptions - if there are no sufficient funds, the withdrawal request will be declined*

***Includes*** *- U16*

***Notes/Issues*** *- This is a child use case of U12*

### *3.3.15 U15: Document Deposit*

***Author –*** *Emma Ude*

***Purpose*** *- To document a deposit made to the club's financial account*

***Requirements Traceability –*** *Traceable to requirement F4*

***Priority*** *- High, contains sensitive information*

***Preconditions*** *- must have access to that specific club's information.*

***Post conditions*** *- The deposit will be documented in the club's financial information*

***Actors*** *– Managers*

***Extends –*** *none*

***Flow of Events***

*Basic Flow - the user will provide information such as, how much they are depositing, how the money was raised, etc. Then the deposit is documented.*
*Alternative Flow - if the user belongs to a specific club(a UCL), they can only deposit funds into that specific club*
*Exceptions - if the funds stated are incorrect, the documentation will be declined.*

***Includes*** *- U16*

***Notes/Issues*** *- This is a child use case of U12*

### *3.3.16 U16: Verify Sufficient Funds*

***Author –*** *Emma Ude*

***Purpose*** *- To verify that the monetary information for withdrawals and deposits are correct*

***Requirements Traceability –*** *Traceable to requirement F4*

***Priority*** *- High, involves sensitive information*

***Preconditions*** *- Must have tried to request a withdrawal or document a deposit*

***Post conditions*** *- If the information is correct, the user's withdrawal or request will be documented*

***Actors*** *– Managers*

***Extends –*** *none*

***Flow of Events***

*Basic Flow - Verifies that the monetary information entered is correct. Then allows the user to proceed*
*Alternative Flow - if the information is not correct, the user cannot proceed with the documentation*
*Exceptions - no exceptions.*

***Includes*** *- none*

***Notes/Issues*** *- This use case is included in U14 and U15*

# 4 Other Non-functional Requirements

## 4.1 Performance Requirements

- *After logging in the user should be redirected to the next page within 3 seconds.*
- *Ensure the application has enough storage to support at least X amount of student membership.*
- *Emails should be sent out to members as soon as an event is created.*
- *A notification will appear on screen if the user attempts to create a new membership for a student whose student ID number is already in the system to reduce redundancy.*

## 4.2 Safety and Security Requirements

- *Passwords are required for the users to gain access to the system to prevent unauthorized access to sensitive information.*
- *For managing memberships and managing finances, the system will verify what information the user can access. For example, UCLs can only access the member information of their specific club, however, the CCMT has administrative access to all information in the system.*
- *When a user tries to request withdrawals or document deposits of funds, the system has to verify that the information entered is accurate, and that there are sufficient funds in the account.*
- *To avoid events from being accidentally canceled, UCLs can only cancel events that they created personally. The CCMT can cancel all events.*

## 4.3 Software Quality Attributes

### 4.3.1 Usability

*The user interface will be created in such a way that it is easy to follow and understand. It will be an embedded software system with buttons that lead to the pages of the system. The system will also use a clear and easy to read font and simple colours to make it easy to follow. Information will also be arranged in a logical manner to make sure it is easy to follow.*

### 4.3.2 Maintainability

*Through the use of abstraction, the system will be divided into different components and then it will be implemented (such as the event management, membership management, etc). This will reduce the interdependency of the different parts and will make it easier to flag and correct errors in a particular system without affecting another component of the system.*

### *4.3.3 Reusability*

*Since this software product is being designed to perform the basic functions of a club event management system, it can be reused for other clubs, or other event functions. With a few changes it can be reused as a wedding planner even.*

# Appendix A - Group Log

## *A.1. Meeting Minutes*

### *A.1.1. January 27, 2022, 5:00 - 6:40pm*

***Location:*** *Online via Zoom*
***Attendees:*** *Maria DeMelo, Emma Ude,Jason Guo, Erfan Razavi*

***Agenda Item 1:*** *Work out a rough draft for the Project Plan*
*Maria googled project plan templates. After some discussion, it was decided that our project plan would outline the different assignment deadlines and the various tasks required for each assignment. Emma was tasked with picking a template and making the document.*

***Agenda Item 2:*** *Figure out what exactly was being asked of us for Project*
*We read through the assignment instructions and noted down some important aspects of it. Jason and Emma pointed out a few things that we need to look out for in the assignment. Some questions about what software we had to use for the assignment and if we needed to use other supplementary software. Maria was tasked with sending the professor an email about the questions that were raised. A shared document was created to write down the things we needed to focus on in this document.*

***Agenda Item 3:*** *Decide what Software to Use For Our Project*
*After some discussion we decided to go with Java as that is what most of us are familiar with. Jason also made suggestions about a database and using HTML for the front end of the application though nothing was finalized in the meeting.*

***Agenda Item 4:*** *Figure out what to do for the Specification Document*
*We googled various templates for the assignment but could not really decide on anything. So it was decided that we would wait for further instruction from the professor.*

### *A.1.2. February 10, 2022, 5:00 - 6:00pm*

***Location:*** *Online via Zoom*
***Attendees:*** *Maria DeMelo, Emma Ude*

***Agenda Item 1:*** *Continue Working on the Specification Document*
*Maria and Emma continued the work on the specification document that had already been created early in the week to finish up in time for the due date.*

## *A.2. Group Activities*

- *Maria emailed the professor to inquire about the assignment instructions*
- *Emma created the Project Plan document and filled it in with the information that was decided on in the meeting.*
- *Maria, Erfan, and Jason made adjustments to the project plan.*
- *Maria made the Specifications document for the assignment and initially filled out Section 1.*
- *Maria sent the Specification document to Emma*
- *Emma shared the specification document with the whole group*
- *Emma worked on the Use Case diagram and its description. Then she sent it to the group for their approval.*
- *Maria worked on Section 2 of the specification document*
- *Emma worked on the rest of Section 3 of the specification document*
- *Emma and Maria worked on Section 4 of the specification document together.*

## *A.3. Additional Communication Information*

*The group's main mode of communication is a Whatsapp Group chat. We talk there regularly between meetings to discuss issues with our work. We also use Whatsapp to pick a time for our meetings. We used this group chat to make the following decisions:*

- *Decide what project we would choose and who would be our contact person.*
- *Make concrete decisions on what Project Management Software to use, we chose Wrike.*
- *Discussions on the layout for our software product.*
- *Talk about the Use Case diagram and other things involved in the specification document.*